In the Claims

Please amend the claims as follows:

B6

1. (Currently Amended) A processor having a changeable architected state, comprising:

instruction memory for storing instructions;

an instruction pipeline, wherein an instruction which passes entirely through the pipeline alters the architected state and wherein the pipeline comprises circuitry for fetching instructions from the instruction memory into the pipeline;

circuitry for storing an annul code having an annul bit corresponding to each instruction of a group of instructions in the pipeline; and

circuitry for preventing one or more selected instructions in the group from altering the architected state in response to the corresponding annul bit of the annul code.

2. (Original) The processor of claim 1:

wherein the instruction pipeline further comprises a plurality of execution units; and

wherein one of more the plurality of execution units receives a corresponding instruction for executing the corresponding instruction in a given clock cycle.

3. (Currently Amended) The processor of claim 2:

wherein each annul bit of the annul code comprises ~~a plurality of bit states~~ either of a first state or a second state;

wherein the circuitry for preventing one or more selected instructions in the group from altering the architected state comprises circuitry for coupling ~~the plurality of bit states~~ annul bits to respective ones of the plurality of execution units;

wherein in response to ~~a bit state~~ an annul bit being a first state the execution unit to which the annul bit ~~state~~ is coupled does not execute the corresponding instruction in the given clock cycle; and

wherein in response to ~~a bit state~~ an annul bit being a second state different than the first state the execution unit to which the annul bit ~~state~~ is coupled does execute the corresponding instruction in the given clock cycle.

B6

4. (Original) The processor of claim 3 wherein the plurality of execution units comprises a load/store unit, a multiply unit, an ALU unit, and a shift unit.

5. (Currently Amended) The processor of claim 3:

wherein the plurality of execution units are operable such that in a given clock cycle an integer number N of the plurality of execution units are scheduled to execute; and

wherein the circuitry for coupling the ~~plurality of bit states~~ annul bits to respective ones of the plurality of execution units comprises circuitry for coupling only the integer number N of the ~~plurality of bit states~~ annul bits to the plurality of execution units which are scheduled to execute in the given clock cycle.

6. (Currently Amended) The processor of claim 5:

wherein the instructions corresponding to the annul code comprise:

a first group of one or more instructions logically arranged after a conditional instruction and to be executed if the condition is satisfied;

a second group of one or more instructions logically arranged after the conditional instruction and to be executed if the condition is not satisfied;

wherein the circuitry for preventing prevents the first group of instructions from altering the architected state in response to corresponding annul bits of the annul code having the first state if the condition is not satisfied; and

wherein the circuitry for preventing prevents the second group of instructions from altering the architected state in response to corresponding annul bits of the annul code having the first state if the condition is satisfied.

7. (Currently Amended) The processor of claim 5:

wherein the instructions corresponding to the annul code comprise instructions corresponding to a software loop scheduled to execute for an integer M number of iterations; and

wherein during a given iteration the circuitry for preventing prevents one or more of the instructions corresponding to the annul bits of the annul code from altering the architected state in response to the annul bits of the annul code and based on a relationship of the given iteration to the integer M number of iterations.

8. (Currently Amended) The processor of claim 3:

wherein the instructions corresponding to the annul code comprise instructions corresponding to software loop scheduled to execute for an integer M number of iterations; and

wherein during a given iteration the circuitry for preventing prevents one or more of the instructions corresponding to the annul bits of the annul code from altering the architected state in response to the annul bits of the annul code and based on a relationship of the given iteration to the integer M number of iterations.

9. (Currently Amended) The processor of claim 3:

wherein the group of instructions corresponding to the annul code comprise:

a first group of one or more instructions logically arranged after a conditional instruction and to be executed if the condition is satisfied;

B6

a second group of one or more instructions logically arranged after the conditional instruction and to be executed if the condition is not satisfied;

wherein the ~~bit states~~ annul bits corresponding to the first group of one or more instructions are set to the first state and the ~~bit states~~ annul bits corresponding to the second group of one or more instructions are set to the second state if the condition is not satisfied; and

wherein the ~~bit states~~ annul bits corresponding to the first group of one or more instructions are set to the second state and the ~~bit states~~ annul bits corresponding to the second group of one or more instructions are set to the first state if the condition is satisfied.

10. (Original) The processor of claim 1 wherein the annul code is generated in response to one or more constant generating instructions.

11. (Original) The processor of claim 1 wherein the annul code is loaded from a memory.

12. (Original) The processor of claim 1 wherein the annul code is an immediate value in an instruction passing through the pipeline.

13. (Original) The processor of claim 1 wherein the annul code comprises 32 bits.

14. (Original) The processor of claim 1 wherein the annul code comprises more than 32 bits.

BC

15. (Original) The processor of claim 1:
wherein the annul code comprises 64 bits; and
wherein the annul code is formed in response to two thirty-two bit values.

16. (Currently Amended) The processor of claim 1:
wherein the annul code is loaded in response to an instruction having a condition predicate;
wherein the annul code comprises a first annul code loaded in response to the condition predicate being satisfied; and
wherein the annul code comprises a second annul code loaded in response to the condition predicate not being satisfied.

17. (Currently Amended) The processor of claim 16:
and further comprising a first data register and a second data register;
wherein the first annul code is stored in the first data register; and
wherein the ~~first~~ second annul code is stored in the second data register.

18. (Currently Amended) The processor of claim 16:
and further comprising a data register;
wherein the first annul code is stored in a first one-half of the data register; and

wherein the second annul code is stored in a second one-half of the data register different from the first one-half.

19. (Original) The processor of claim 1:

wherein an annul instruction passing through the pipeline specifies an integer N; and

wherein the annul code is formed in response to the integer value N such that the circuitry for preventing prevents N successive instructions in the pipeline from altering the architected state.

20. (Currently Amended) The processor of claim 1:

and further comprising a register;

wherein the register stores the annul code which comprises a set of annul bits having a first logical value and a set of annul bits having a second logical value;

wherein the annul code is loaded in response to an instruction having a condition predicate;

wherein the circuitry for preventing prevents instructions corresponding to ~~the~~ annul bits having a first logical value from altering the architected state in response to the condition predicate being satisfied; and

wherein the circuitry for preventing prevents instructions corresponding to ~~the~~ annul bits having a second logical value opposite the first logical state from altering the architected state in response to the condition predicate not being satisfied.

21. (Original) The processor of claim 1 and further comprising circuitry for storing a portion of the annul code in response to receipt of an interrupt.

22. (New) A method of data processing comprising the steps of:

identifying a group of instructions including a tree of a plurality of conditional branch instructions;

for each conditional branch instruction within the group of instructions

B6

forming a first annul code having an annul bit corresponding to each instruction following the conditional branch instruction, the annul bit having a first logical state for instructions following the detected conditional branch instruction executed if a condition of the conditional branch instruction is satisfied and a second logical state opposite to the first logical state for instructions following the detected conditional branch instruction executed if the condition of the conditional branch instruction is not satisfied,

forming a second annul code having an annul bit corresponding to each instruction following the conditional branch instruction, the annul bit having the second logical state for instructions following the detected conditional branch instruction executed if a condition of the conditional branch instruction is satisfied and the first logical state for instructions following the detected conditional branch instruction executed if the condition of the conditional branch instruction is not satisfied;

upon execution of the group of instructions

detecting each conditional branch instruction,

evaluating the condition of the conditional branch instruction,

loading the corresponding first annul code if the condition of the conditional branch instruction is satisfied,

loading the corresponding second annul code if the condition of the conditional branch instruction is not satisfied,

executing each instruction following the conditional branch instruction if the corresponding annul bit of the corresponding annul code has the first state, and

not executing each instruction following the conditional branch instruction if the corresponding annul bit of the corresponding annul code has the second state.